Code-A-Ton 2024

Features supported:

1. Lyrics sentiment classification
    a. Unsupervised learning method
        i. K-Means clustering
    b. Supervised learning method
        i. Fully connected neural network.
2. Lyrics Generation
    a. Word2vec model implementation
        i. Without template
        ii. With template
3. Experiment tracking using Tensorboard.
4. Gradio for interacting with the trained models.

Lyrics sentiment classification:

- As the initial assessment was sent with the million unlabeled Spotify dataset the first idea that came to mind was to use clustering techniques to group the songs together.
- But clustering techniques don't usually work well on texts, due to the complexity of the language. But as I had worked with word embeddings before and as word embeddings are better in capturing the word context and structure of the language, I thought to use word embeddings for all the lyrics from "word2vec" model and then perform clustering on top of these embeddings.

Word embeddings

- I am using genism python library to load the pre-trained word2vec embeddings. Gensim comes with a wide range of embeddings dataset, for this project I'm using "word2vec-google-news-300" which contains embeddings from the word2vec model trained on google news, each embedding is of 300 dimensions vector(1x300).
- Before moving on to K-means clustering I wanted to see if I could use just the embeddings to predict the sentiment of the song using the similarity between the embedding vectors.
- Embedding vectors have a unique property that if a word occurs in similar "context" as the other word then the cosine distance between these two embeddings is smaller i.e they are closer to each other.
    - For example : The embeddings for the word "king" is closer to "man", whereas the embeddings for the word "queen" is closer to "women".
    - The embeddings also support vector operations.

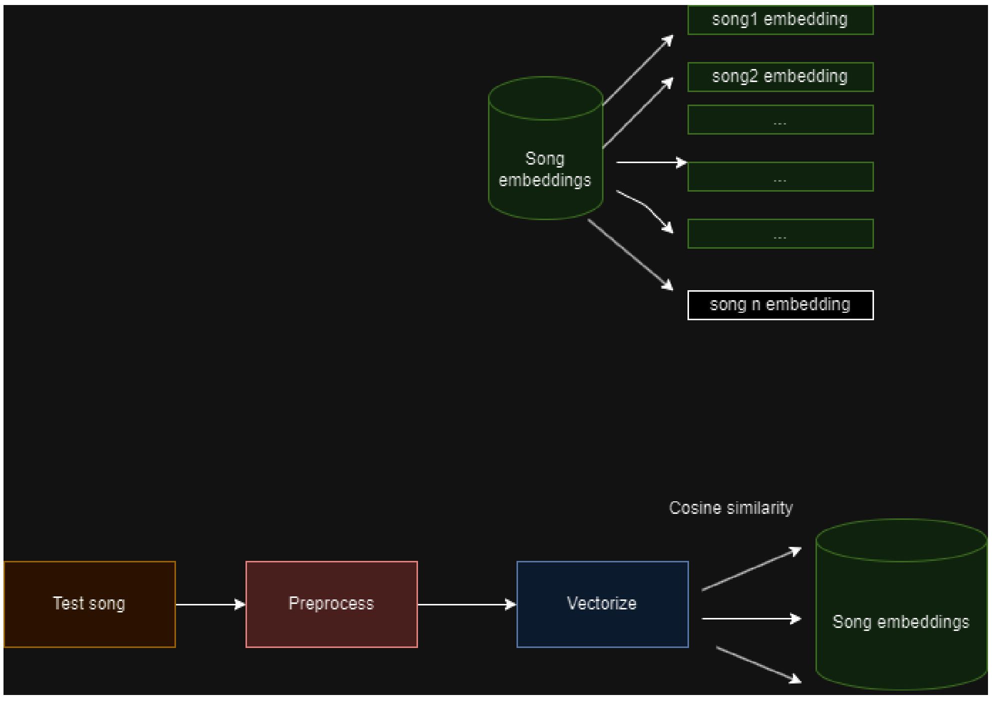
- To try out this method I searched for labelled datasets for song lyrics, and I came across this dataset. It is a multi-labeled dataset, it has the following sentiment categories – Amazement, Calmness, Joyful activation, Nostalgia, Power, Sadness, Solemnity, Tenderness, Tension

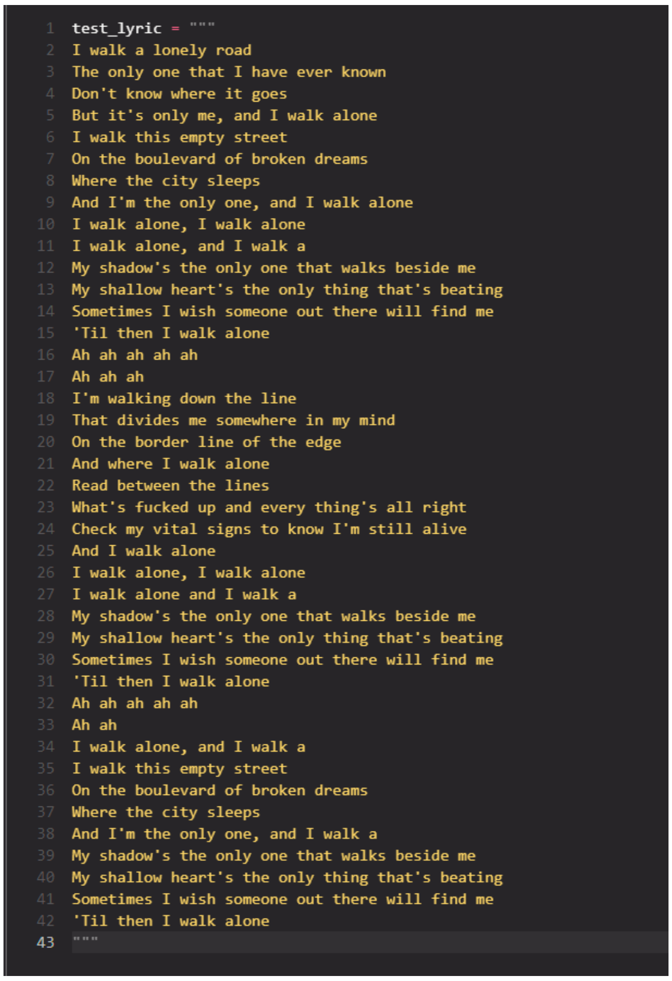
Data preprocessing[1]

- To convert the song lyrics into embeddings I used the following preprocessing technique.
    - Lower case the text
    - Tokenize the sentences using NLTK.
    - Remove stop words from the tokens list, filter out non alphabetic tokens.
    - Lemmatize the tokens.
- For creating an embedding for the song lyrics, I calculated the embeddings for each token, do this for every token in the lyrics, and then take the sum of all the token embeddings.
- Do the above for every song, and then normalize the embeddings to get the unit variance.

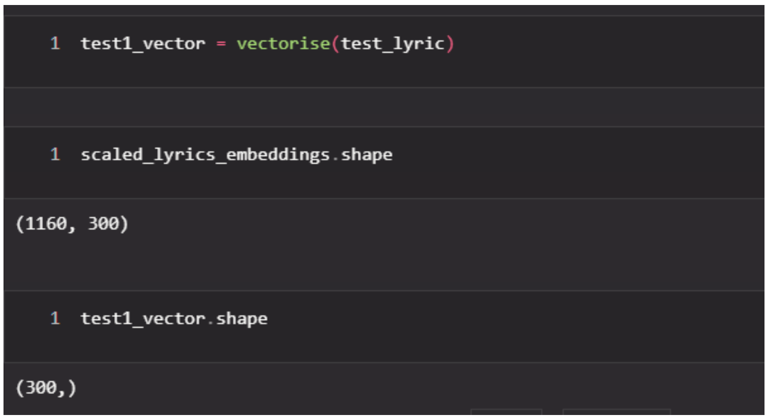
Prediction

- Given a song lyric, I calculate the vector embeddings as mentioned above.
- Then calculate the "cosine similarity" of this vector with matrix of embeddings which I had created in the pre-processing step. This gives a similarity score for each song present in our corpus.
- Now select the song which has the highest similarity score, and the label belonging to this song is the label assigned to the test song.





```
test_lyric = """
I walk a lonely road
The only one that I have ever known
Don't know where it goes
But it's only me, and I walk alone
I walk this empty street
On the boulevard of broken dreams
Where the city sleeps
And I'm the only one, and I walk alone
I walk alone, I walk alone
I walk alone, and I walk a
My shadow's the only one that walks beside me
My shallow heart's the only thing that's beating
Sometimes I wish someone out there will find me
'Til then I walk alone
Ah ah ah ah ah
Ah ah ah
I'm walking down the line
That divides me somewhere in my mind
On the border line of the edge
And where I walk alone
Read between the lines
What's fucked up and every thing's all right
Check my vital signs to know I'm still alive
And I walk alone
I walk alone, I walk alone
I walk alone and I walk a
My shadow's the only one that walks beside me
My shallow heart's the only thing that's beating
Sometimes I wish someone out there will find me
'Til then I walk alone
Ah ah ah ah ah
Ah ah
I walk alone, and I walk a
I walk this empty street
On the boulevard of broken dreams
Where the city sleeps
And I'm the only one, and I walk a
My shadow's the only one that walks beside me
My shallow heart's the only thing that's beating
Sometimes I wish someone out there will find me
'Til then I walk alone
"""
```



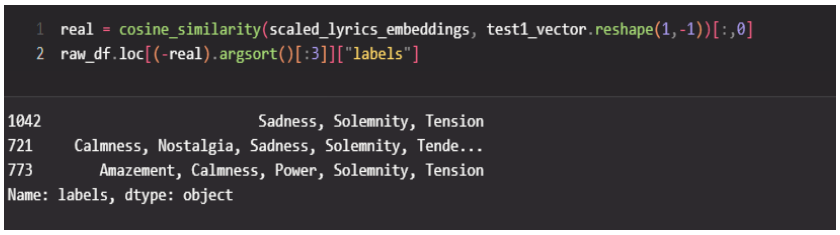
```
test1_vector = vectorise(test_lyric)
```

```
scaled_lyrics_embeddings.shape
```

```
(1160, 300)
```

```
test1_vector.shape
```

```
(300,)
```



```
real = cosine_similarity(scaled_lyrics_embeddings, test1_vector.reshape(1,-1))[:,0]
raw_df.loc[(-real).argsort()[:3]]["labels"]
```

```
1042                          Sadness, Solemnity, Tension
721     Calmness, Nostalgia, Sadness, Solemnity, Tende...
773          Amazement, Calmness, Power, Solemnity, Tension
Name: labels, dtype: object
```

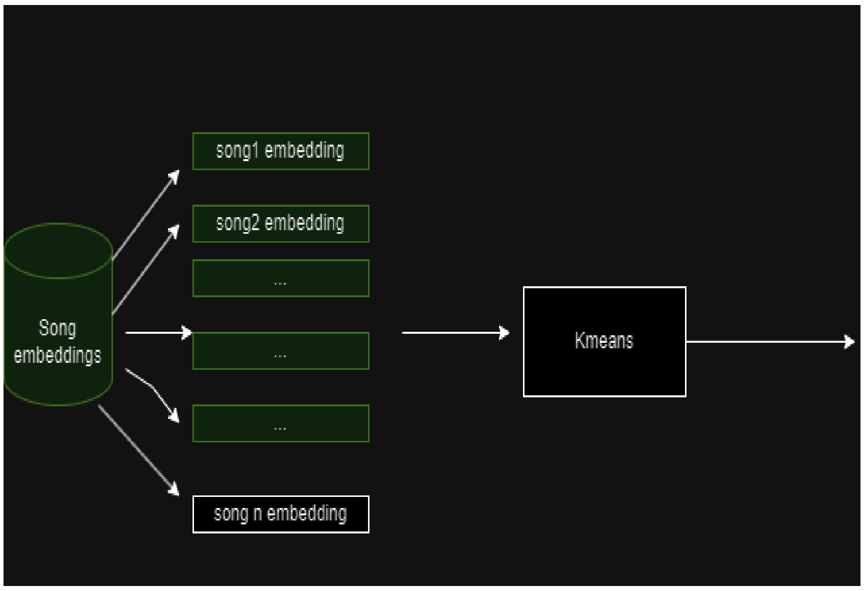
Experiment

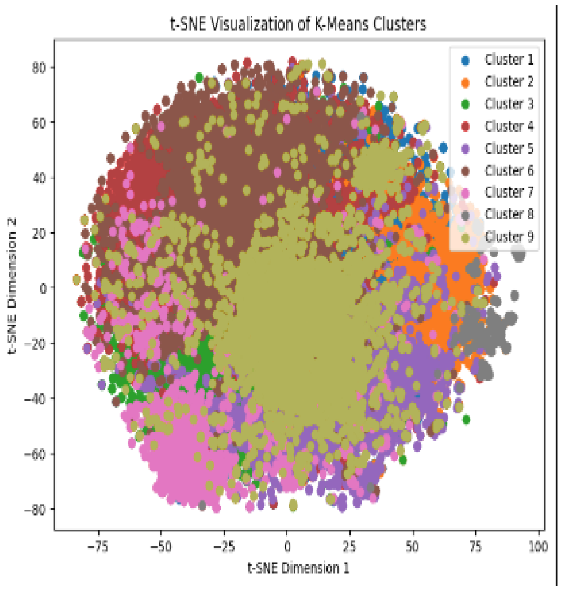
- For predicting the label for a song, now we need to store all the vectors of the songs in our corpus. This wouldn't scale as more the songs more the number of embeddings.
- For this purpose, I thought of combining the vectors of all the songs belonging to a particular emotion. For example, if there are 10 songs belonging to the emotion "Sadness" then I combine all the vectors to a single vector.
- Now I will be left with 9 embedding vectors which I can use for predicting the labels for new songs by computing the cosine similarity.
- After computing this embedding matrix of size (9 x 300), I kind of stumbled upon the fact that the combined embeddings for "Calmness" and "Sadness", "Power" and "Tension", "Amazement" and "Solemnity" are nearly identical. Through this observation we can conclude that songs belonging to one category can be equal to the other category.

Kmeans

- As discussed earlier, now I have established the process to convert a song lyric from textual representation to embedding representation.
- Using this emebeddings data I can perform clustering with the help of clustering techniques such as Kmeans, DBSCAN etc.
- The reason I chose Kmeans is that it's easy to implement as there was constraint not to use any deep learning library.
- Now I employ the same approach as I did in the previous embedding method[1].

Process

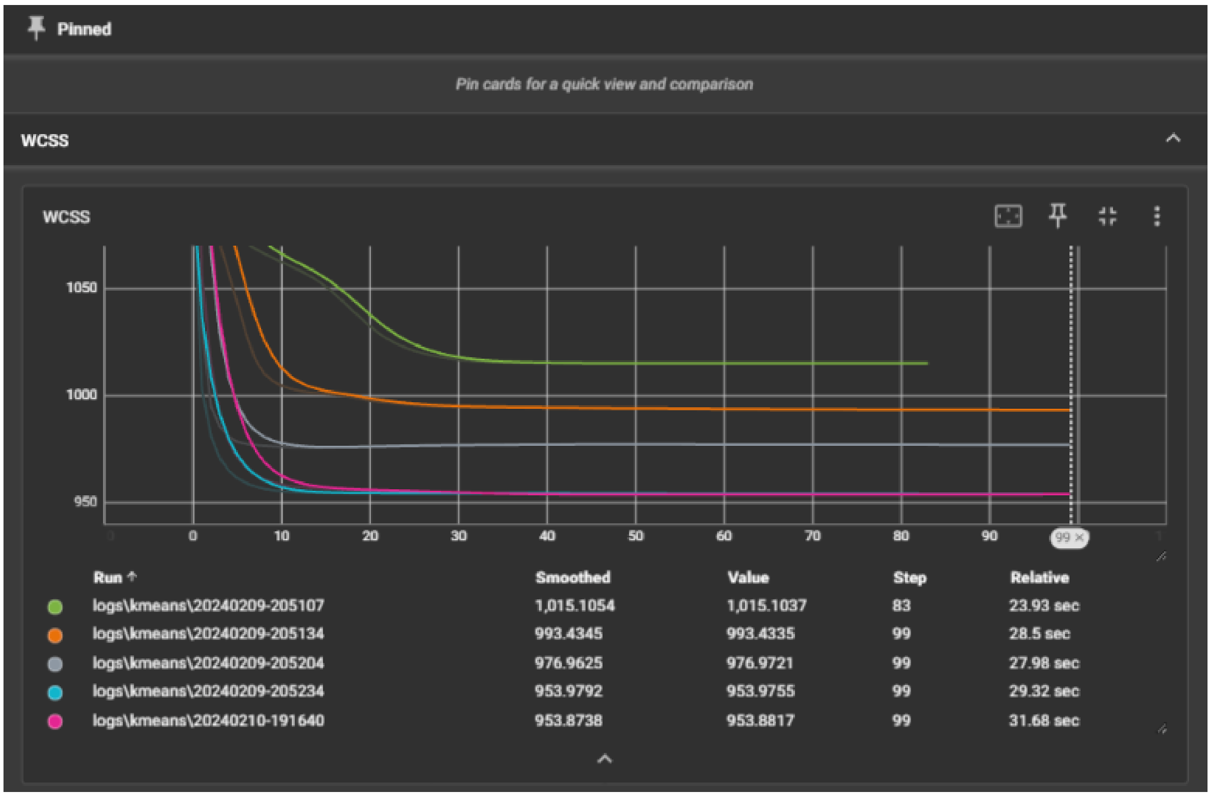




- Kmeans is a simple algorithm, and it's easy to implement but tough to decide when to stop the training and what each cluster represents.
- Kmeans is non-parametric model i.e it does not have any parameters to learn during the training stage. The hyperparameters are the number of clusters K.
- For measuring the performance of the algorithm I'm using WCSS – Within Cluster Sum of Squares.
- One change I've made is, usually we use Euclidean distance in measuring the distance between the centroids and the data point, here I'm using cosine distance as we are dealing with the embeddings.
- I stop the training when the centroids don't change from the previous step.

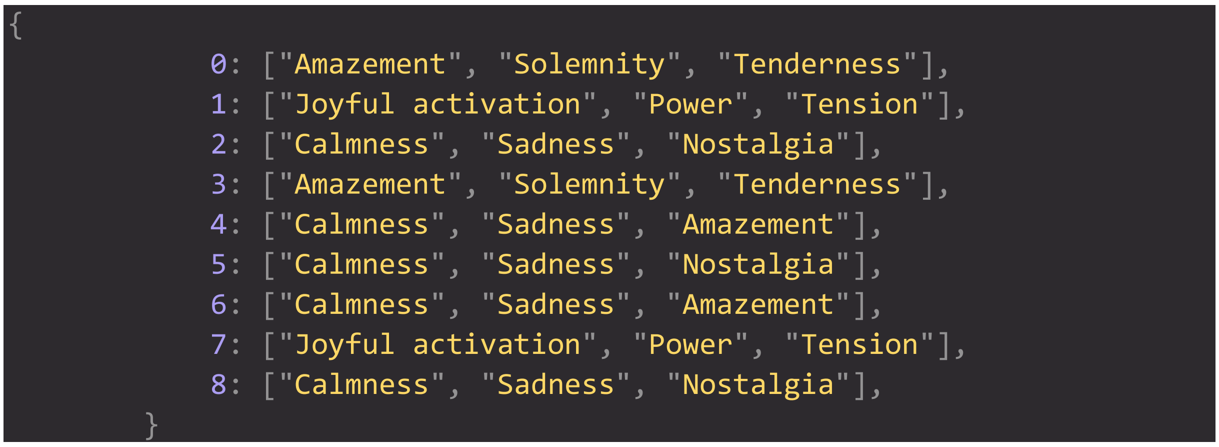
Experiments

- As Kmeans doesn't have many hyperparameters to tune, I tried out some values of K and selected the right metric based on WCSS distance.
- All the experiments are tracked using Tensorboard.



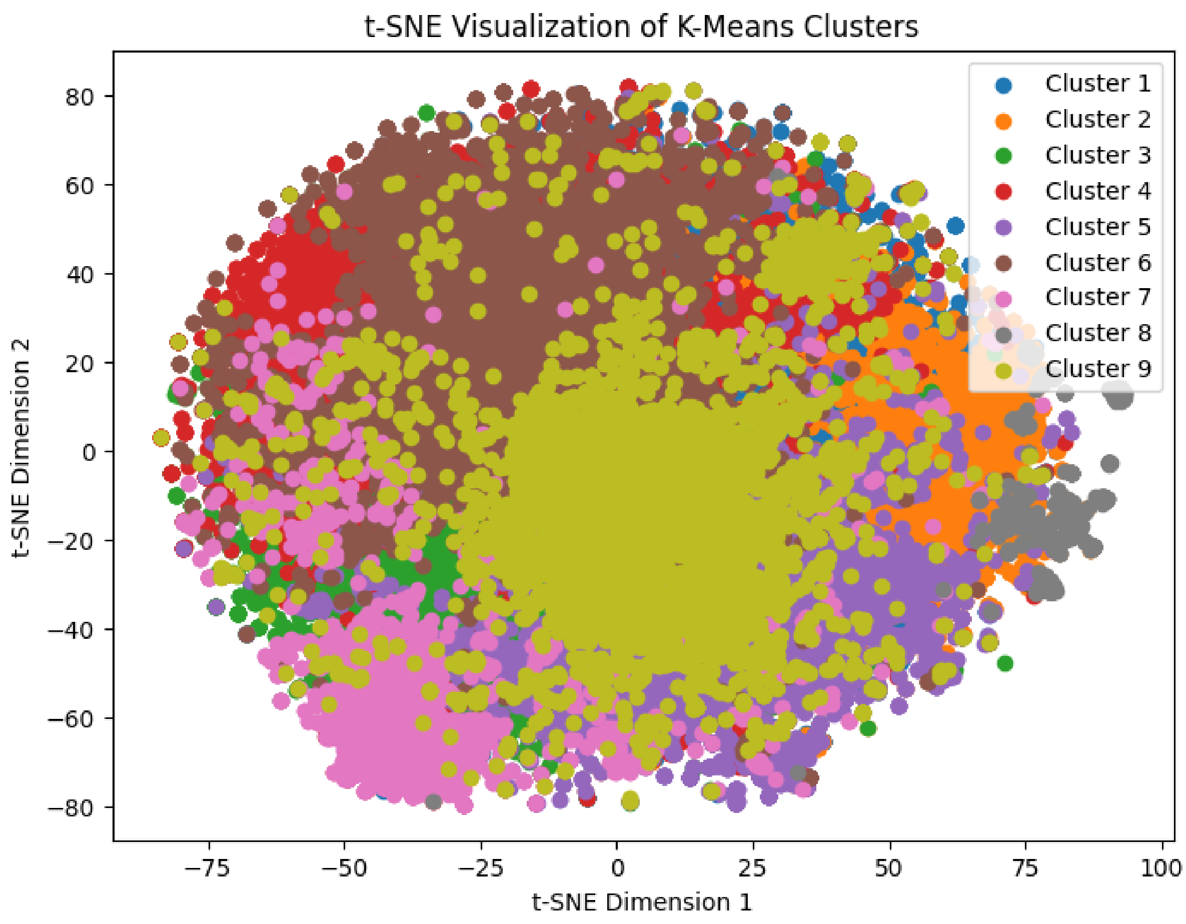
	Run ↑	Smoothed	Value	Step	Relative
🟢	logs\kmeans\20240209-205107	1,015.1054	1,015.1037	83	23.93 sec
🟠	logs\kmeans\20240209-205134	993.4345	993.4335	99	28.5 sec
⚪	logs\kmeans\20240209-205204	976.9625	976.9721	99	27.98 sec
🔵	logs\kmeans\20240209-205234	953.9792	953.9755	99	29.32 sec
🟣	logs\kmeans\20240210-191640	953.8738	953.8817	99	31.68 sec

- The best **WCSS score is 953.88 for K = 9**.
- For assigning the labels for each centroid and it's datapoints I made use of embeddings from the previous approach (embedding approach) as we know for each label each of the 9 embeddings belong to. When I did the cosine similarity with these embeddings, I got the below labels for the centroids.
- After training this K-Means I have used it to predict the labels for the given Spotify dataset.



```
{
            0: ["Amazement", "Solemnity", "Tenderness"],
            1: ["Joyful activation", "Power", "Tension"],
            2: ["Calmness", "Sadness", "Nostalgia"],
            3: ["Amazement", "Solemnity", "Tenderness"],
            4: ["Calmness", "Sadness", "Amazement"],
            5: ["Calmness", "Sadness", "Nostalgia"],
            6: ["Calmness", "Sadness", "Amazement"],
            7: ["Joyful activation", "Power", "Tension"],
            8: ["Calmness", "Sadness", "Nostalgia"],
        }
```

TSNE visualization of the clusters



- As the centroids are of 300-dimension vectors we can’t easily visualize the clusters without doing dimensionality reduction.
- For dimensionality reduction we can use techniques such as PCA, Singular value decomposition, but these two are slow compared to T-SNE.
- The above figure shows the visualization of our clusters in 2 T-SNE dimensions.

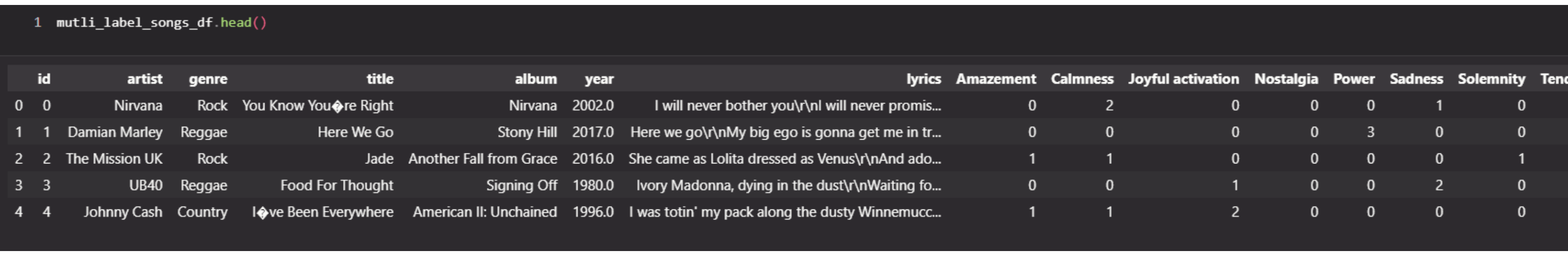
Fully connected neural network

- Supervised learning methods are preferred over unsupervised learning methods in classification/prediction tasks.
- I therefore searched for labelled datasets containing lyrics and sentiment labels, I came across these datasets – Edmonds dance dataset and lyrics emotion detection dataset. Edmonds dataset has also some tweets dataset which we will use to extend our dataset.

Dataset preprocessing

- All labels in Edmonds dance dataset and lyrics emotion dataset are not similar. But some labels do match, therefore we need to do some pre-processing.

- I have used Edmonds dataset for testing purposes and combined the lyrics emotion dataset along with tweets dataset to create a training dataset.
  These are the steps involved in the preparation of the dataset.
  1. In lyrics emotion dataset there are separate columns for each labels, but they are not one hot encoded instead they are numbered like 1,2,3…

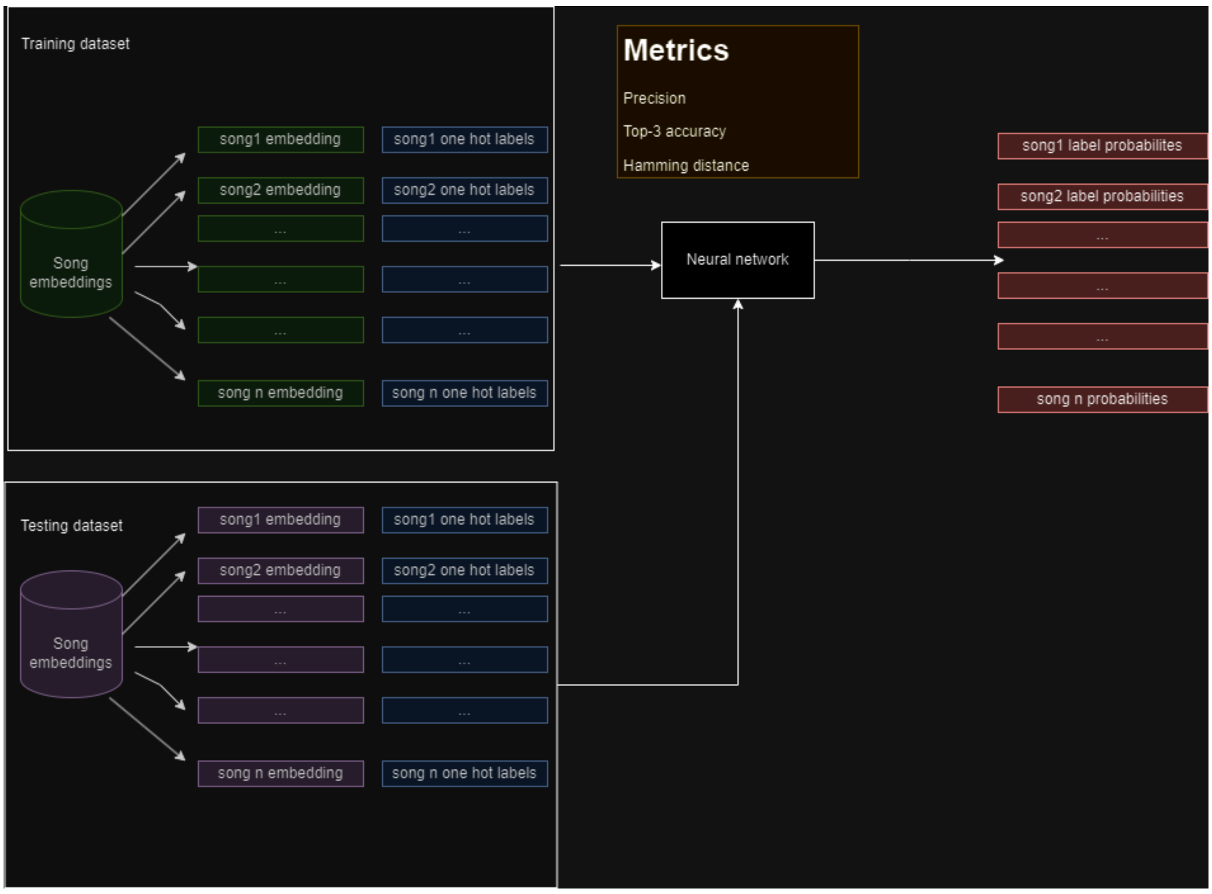


```
mutli_label_songs_df.head()
```

	id	artist	genre	title	album	year	lyrics	Amazement	Calmness	Joyful activation	Nostalgia	Power	Sadness	Solemnity	Tend
0	0	Nirvana	Rock	You Know You�re Right	Nirvana	2002.0	I will never bother you\r\nI will never promis...	0	2	0	0	0	1	0	
1	1	Damian Marley	Reggae	Here We Go	Stony Hill	2017.0	Here we go\r\nMy big ego is gonna get me in tr...	0	0	0	0	3	0	0	
2	2	The Mission UK	Rock	Jade	Another Fall from Grace	2016.0	She came as Lolita dressed as Venus\r\nAnd ado...	1	1	0	0	0	0	1	
3	3	UB40	Reggae	Food For Thought	Signing Off	1980.0	Ivory Madonna, dying in the dust\r\nWaiting fo...	0	0	1	0	0	2	0	
4	4	Johnny Cash	Country	I�ve Been Everywhere	American II: Unchained	1996.0	I was totin' my pack along the dusty Winnemucc...	1	1	2	0	0	0	0	

  2. So, I converted these columns to be one hot encoding.
  3. I combined Nostalgia and Sadness to one column to be sadness and then dropped Nostalgia column.
  4. I dropped Solemnity from the dataset as it is not like any of the labels in the test dataset.
  5. As the test dataset contains “Disgust” which is not present in the training dataset, I took the “Disgust” data from the tweets dataset and created a new feature column in the lyric’s emotion dataset. I set 1 to Disgust column for the tweets and 0 for all other previously songs.

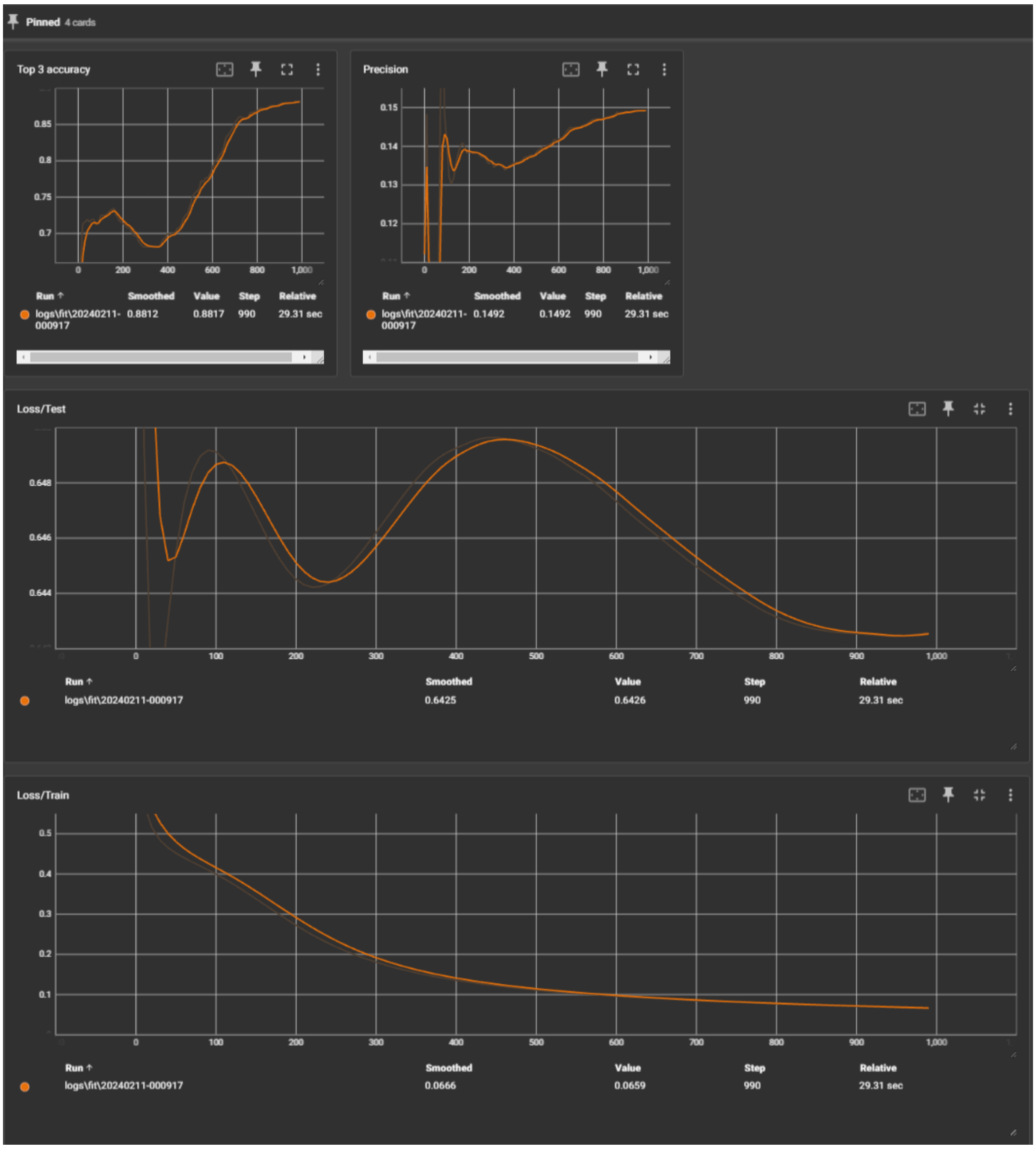
Training



- The preprocessing of lyrics data is like what we have done in the previous two methods.
- The neural network consists of one hidden layer with non-linear activation. The last layer consists of **sigmoid** activations (I'm not using **SoftMax** as the last layer activation as we need to independently predict the class labels), binary cross entropy is used as the loss function.
- For faster convergence I've used **Glorot/Xavier** weight initialization. Compared to random initialization Glorot initialization worked really well.
- The hidden layer uses RELU (rectified liner unit) as the activation function.
- I have used top3 accuracy, Hamming loss and precision for measuring the performance of the model.
- The input to the neural network is the embeddings representation of the lyrics.
- I have used batch gradient descent as the optimization algorithm.
- After training this neural network I have used it to predict the labels for the given Spotify dataset.

Experiment

- Every experiment is tracked using Tensorboard.
- The hyperparameters involved in this neural network are – number of hidden neurons, learning rate, hidden layer activation, batch size.
- The best set of hyperparameters are **hidden neurons – 128, epochs = 1000, learning rate – 0.01, hidden layer activation – relu, batch size – 32. Test error** 0.64, **Precision** 0.149, **top3metric** 0.89, **Hamming loss** 0.33.
- I believe that this method still has the potential to improve by increasing the depth of the neural network.
- Compared to the other two approaches, this method performed better, this is just the subjective analysis of few songs, but there is no benchmarking to prove as I couldn't test the methods on the same dataset due to constraint of time.



Top 3 accuracy

Run ↑	Smoothed	Value	Step	Relative
● logs\fit\20240211-000917	0.8812	0.8817	990	29.31 sec

Precision

Run ↑	Smoothed	Value	Step	Relative
● logs\fit\20240211-000917	0.1492	0.1492	990	29.31 sec

Loss/Test

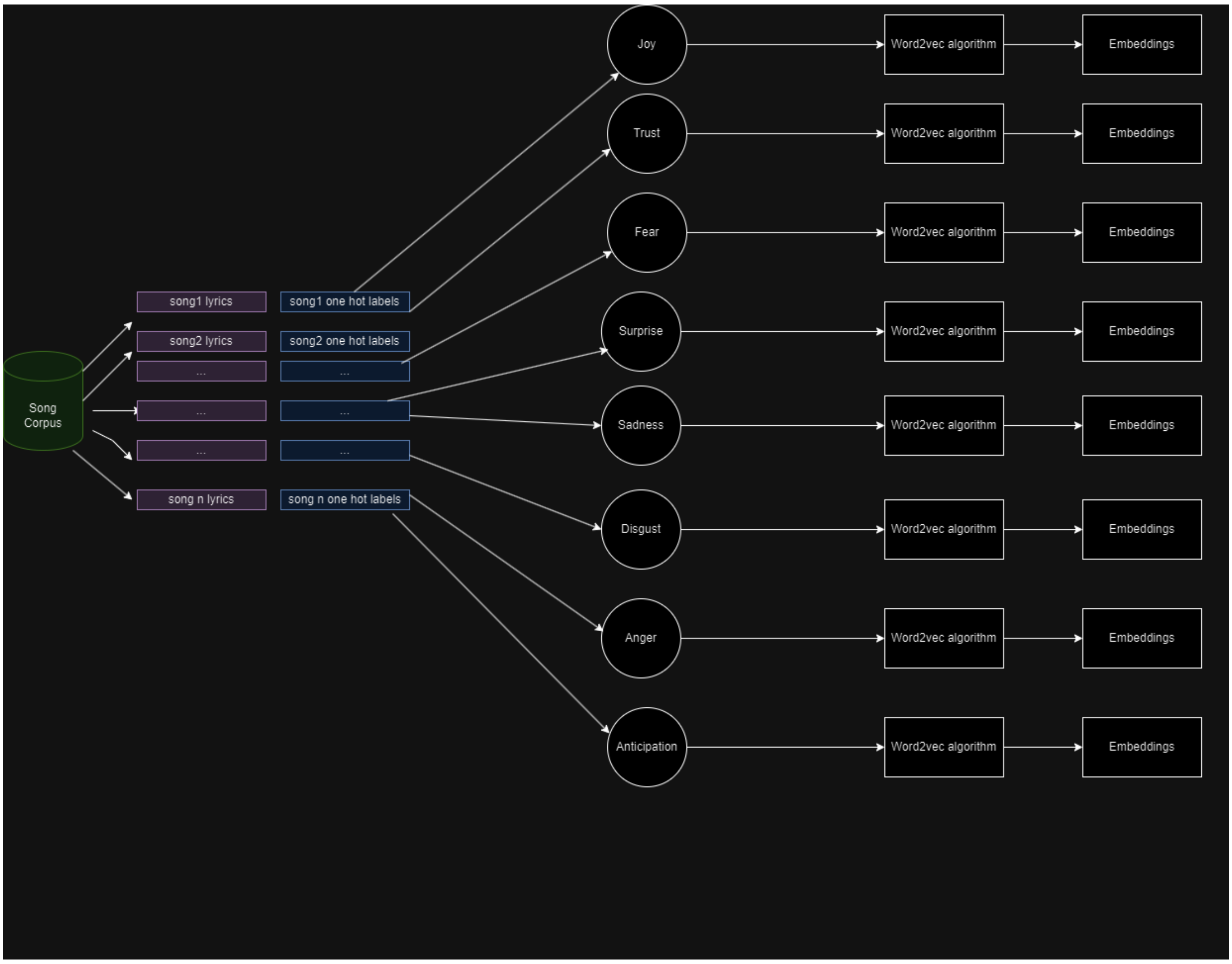
Run ↑	Smoothed	Value	Step	Relative
● logs\fit\20240211-000917	0.6425	0.6426	990	29.31 sec

Loss/Train

Run ↑	Smoothed	Value	Step	Relative
● logs\fit\20240211-000917	0.0666	0.0659	990	29.31 sec

Lyrics Generation

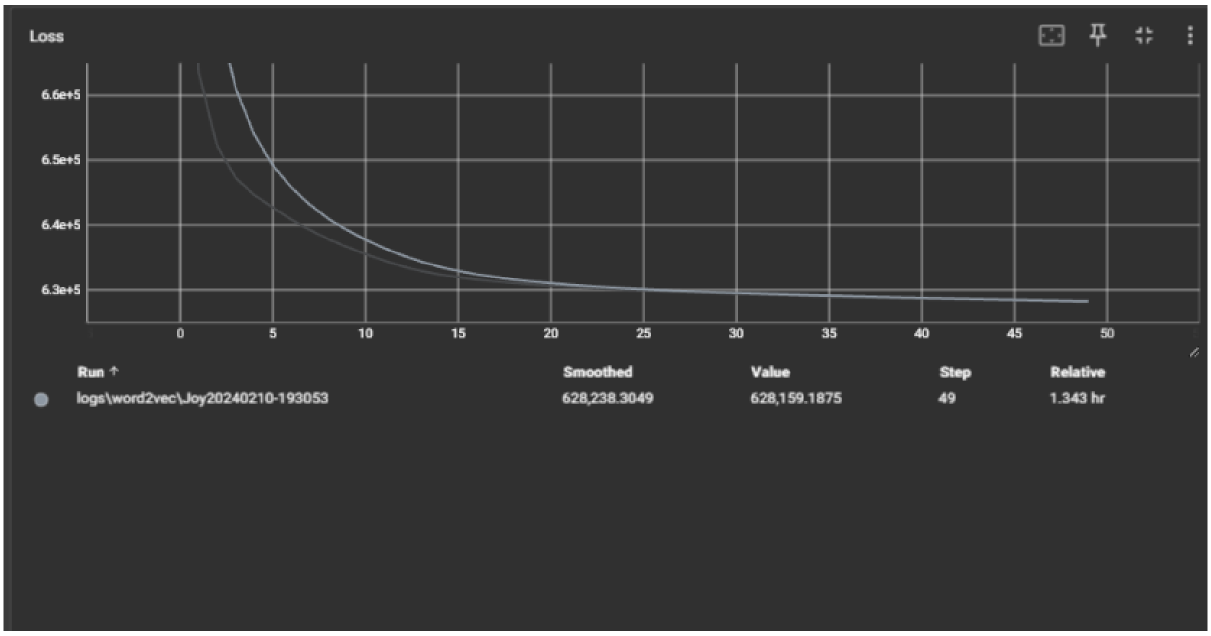
For lyrics generation, I have implemented word2vec algorithm to detect the surrounding/outer words given the center word. I have employed two approaches which are distinguished by the way the words are ordered.



Training

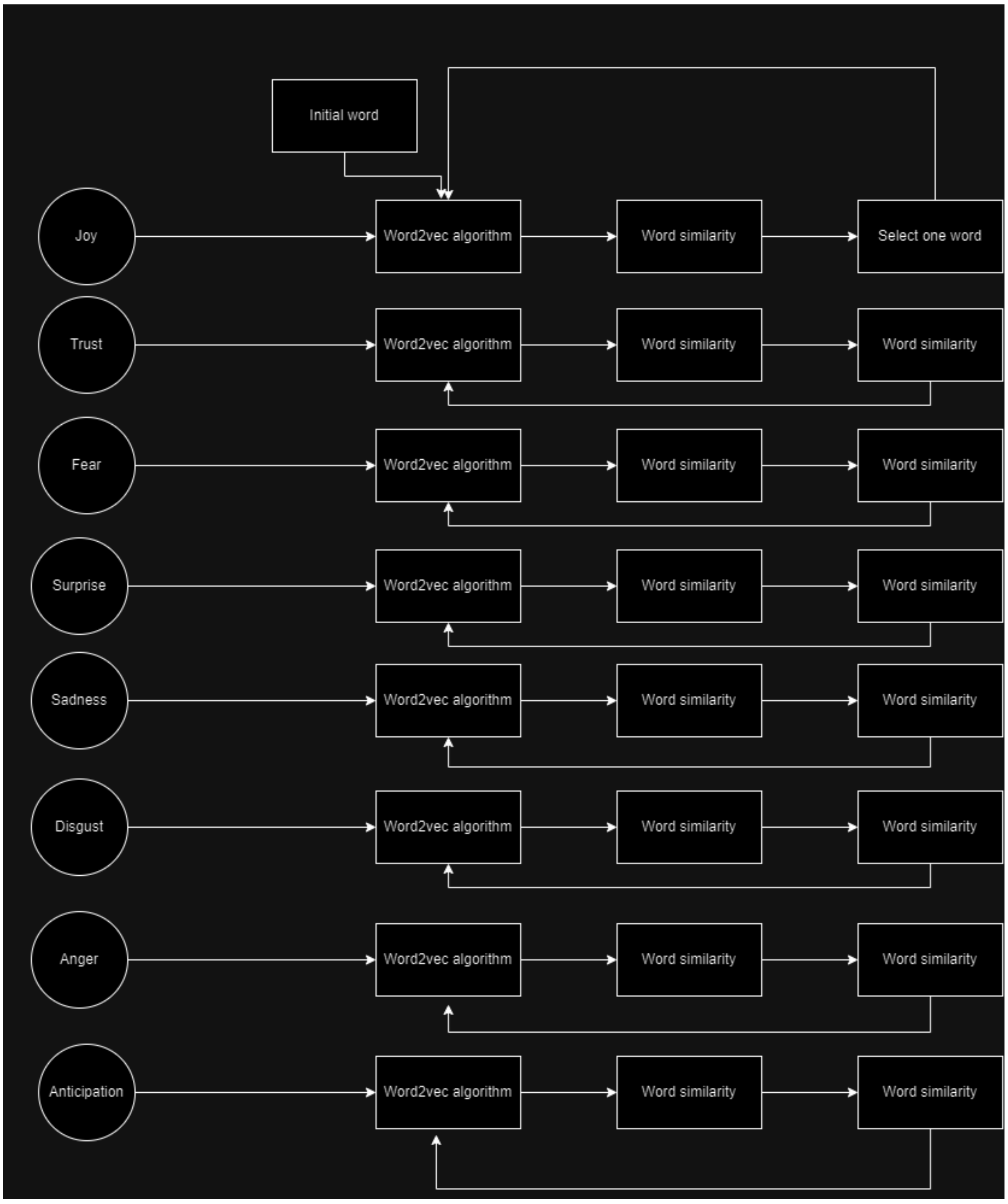
- The songs in the Edmonds dataset are filtered out into different groups based on their emotion labels. The groups are not exclusive, i.e. a song can be more than one group, as there are multiple emotions in a song.
- Individual word2vec model is trained on each emotion cluster and the word2vec model produces embeddings for each token in the cluster. Now we have a mapping for each token and its embeddings from the algorithm.
- The embeddings produced by the model have learned the semantics belonging to each emotion. This helps in generating relevant words for generating the lyrics.
- I couldn’t train on all the songs present in the cluster as it would take lot of time, so I trained the algorithm on 50 songs in each cluster.

- The model's training is tracked on Tensorboard, I finetuned the hyperparameters for emotion "Joy" and the best hyperparameters are - **hidden layer neurons** – 3, **window size** – 3, **epochs** – 50, **learning_rate** – 0.01. The cross entropy loss at the end of 50 epochs is – 6.28e5.



Run ↑	Smoothed	Value	Step	Relative
logs\word2vec\Joy20240210-193053	628,238.3049	628,159.1875	49	1.343 hr

Lyrics generation without template –



- The first technique I used to generate the lyrics given the emotion is –
  - Select the word2vec (w2v) model based on the emotion, then feed the word2vec (w2v) model with an initial word.
  - As the w2v is trained, get the list of words which are semantically nearer to the initial word. Then choose one of the words and again feed the selected word to the w2v model and get the list of words, and repeat this process until you have the required number of words.

   - To make sure there are a smaller number of repeated words, maintain a queue, which stores the last 5 words generated.
   - Sample lyrics generated for emotion “Joy” and initial word “I”
      - ***“ i w- all might often apart . if crazy romance . , alright. shawty how we god asleep kelis . stronger no louie . i offered kinda im hear , . drunk joke headband is gisele . owned me freak ok about would you are lucky could imitate . whatever feeling wo again try causing something > crew dancefloor second . . sun bone call , should , ai taste knew , , murder gim they gim they gim they , gim . they gim they , gim they gim they gim they gim they gim they gim they gim “.***
   - The first thing that we observe is that there is no structure in the sentences. And it seems gibberish, some words appear to resonate the emotion provided.
   - To solve this issue I came up with another methodology which follows a certain structure while forming the sentences.

Lyrics generation with template –

- As there is a lack of structure in the previous method I tried to break up the songs into verses, chorus, bridge and outro. I took the song named “Boulevard of Broken Dreams” by “Green Day”, and used prompt engineering on ChatGPT to break the song into verses, choruses, bridge and outro.
- Then for each song section I used NLTK to tokenize and assign each token a “Part of speech” (POS) tag. And using this information, I came up with this template for the songs to be generated

```
{
  "verse1": "Pronoun, Verb, Determiner, Adjective,
Noun,<br>,Determiner, Adjective, Noun, Determiner, Pronoun, Verb,
Adverb, Verb,<br>,Verb, Adverb, Verb, Adverb, Pronoun,
Verb,<br>,Conjunction, Pronoun, Verb, TO, Pronoun, Conjunction,
Pronoun, Verb, Noun, Preposition, Pronoun, Conjunction, Pronoun, Verb,
Adjective",
  "verse2":
"Pronoun,Verb,Adjective,Pronoun,Conjunction,Verb,Adjective,Pronoun,Verb
,Pronoun,Adjective,<br>,Pronoun,Adjective,Noun,Verb,Determiner,Adjectiv
e,Noun,Pronoun,Verb,Adjective,<br>,Adverb,Pronoun,Verb,Noun,Adverb,Adve
rb,Verb,Pronoun,Conjunction,Adverb,Pronoun,Verb,Adjective,<br>,Interjec
tion,Interjection,Interjection,Interjection,Interjection,Pronoun,Verb,D
eterminer, Adjective",
  "chorus1":
"Adverb,Pronoun,Verb,Noun,Adverb,Verb,Pronoun,Conjunction,Adverb,Pronou
n,Verb,Adjective,Interjection,Interjection,Interjection, Adjective",
  "outro":
"Determiner,Noun,Verb,Determiner,Adjective,Noun,Pronoun,Verb,Prepositio
n,Pronoun,Adjective,<br>,Determiner,Adjective,Noun,Verb,Determiner,Adje
ctive,Noun,Pronoun,Verb,Adjective,<br>,Adverb,Pronoun,Verb,Noun,Adverb,
Verb,Pronoun,Conjunction,Adverb,Pronoun,Verb,Adjective"
}
```

- I also created a mapping for every token in the corpus to it's respective POS tag, so that I can choose the required tokens based on the POS tag required according to the above template.
- Whenever I need to generate lyrics, I make use of the above template and with my corpus of tokens and their POS tags corresponding to each emotion.
- Example lyrics generated using the above template – emotion "Sadness"

it need no wrong hand
all other clap whatever they wonder too hated
gin maybe call why they pateks
but he remains to you ,or him prove soon

him hey every philipp change
on every blue about wanting affection
when this need hand
or we wonder some warm two ,or she add always

we burn holding off both mind
this monica self quite for my life
past some please ,but when it deserve soon

him prove every philipp change
on every on like ha louis
why a rack tag
and we fine another use three ,or self look oh

our tie till all wrong hand whatever dry around self
my direct life clap this creep luck which seems hope
very ,they clap life past still should reliever he
me make just

- We can observe that the structure of the sentences are more clearer, and can make sense out of it. But this method still has flaws has the semantics doesn't make sense.
- I tried to improve this methodology using fine grained template like below

```
{
  "verse1": "personal_pronoun, verb_present_not_third_person_singular,
determiner, adjective_large, noun_singular,<br>,determiner,
adjective_large, noun_singular, wh_determiner, personal_pronoun,
verb_present_not_third_person_singular, adverb,
verb_past_participle,<br>,verb_present_not_third_person_singular,
Adverb, verb, wh_adverb,
personal_pronoun,verb_present_third_person_singular,<br>,coordinating_c
onjunction, personal_pronoun,verb_present_third_person_singular,
infinite_marker, personal_pronoun,<comma>,coordinating_conjunction,
personal_pronoun,verb_present_not_third_person_singular, adverb",

  "chorus1":
"personal_pronoun,verb_present_not_third_person_singular,determiner,adj
ective_large,noun_singular,<br>,preposition_subordinating_conjunction,d
eterminer,proper_noun_singular,preposition_subordinating_conjunction,pr
oper_noun_singular,proper_noun_singular,<br>,wh_adverb,determiner,noun_
singular,noun_singular,<br>,coordinating_conjunction,personal_pronoun,
verb_present_not_third_person_singular,determiner,adjective_large,cardi
nal_digit,<comma>,coordinating_conjunction,personal_pronoun,verb_presen
t_not_third_person_singular,adverb,<br>",

  "bridge":
"personal_pronoun,verb_present_not_third_person_singular,verb_gerund,pa
rticle,determiner,noun_singular,<br>,determiner,verb_present_third_pers
on_singular,personal_pronoun,adverb,preposition_subordinating_conjuncti
on,possessive_pronoun,noun_singular,<br>,preposition_subordinating_conj
unction,determiner,noun_singular,<comma>,coordinating_conjunction,wh_ad
verb,personal_pronoun,verb_present_not_third_person_singular,adverb",

  "chorus2":
"personal_pronoun,verb_present_not_third_person_singular,determiner,adj
ective_large,noun_singular,<br>,preposition_subordinating_conjunction,d
eterminer,proper_noun_singular,preposition_subordinating_conjunction,pr
oper_noun_singular,proper_noun_singular,<br>,wh_adverb,determiner,noun_
singular,noun_singular,<br>,coordinating_conjunction,personal_pronoun,
verb_present_not_third_person_singular,determiner,adjective_large,cardi
nal_digit,<comma>,coordinating_conjunction,personal_pronoun,verb_presen
t_not_third_person_singular,adverb,<br>",

  "outro":
"possessive_pronoun,adjective_large,possessive_ending,determiner,adject
ive_large,noun_singular,wh_determiner,verb_present_third_person_singula
r,preposition_subordinating_conjunction,personal_pronoun,<br>,possessiv
e_pronoun,adjective_large,noun_singular,possessive_ending,determiner,ad
jective_large,noun_singular,wh_determiner,verb_present_third_person_sin
gular,noun_singular,<br>,adverb,<comma>,personal_pronoun,verb_present_n
ot_third_person_singular,noun_singular,preposition_subordinating_conjun
ction,adverb,modal,verb,personal_pronoun,<br>,personal_pronoun,verb_pre
sent_not_third_person_singular,adverb"
}
```

- The sample lyrics generated using this template is as follows, emotion – Sadness

what remind alone sad girl
remind i step face high talk face high
bring be dancin where bring nobody
where face high look tragedy ,left it running sleep

going here fast same hand
somehow knocking somehow knocking here sleep
everything bad mistake dirty
amazing it nothing to left door ,sleep it running sleep

running it going dirty mistake everything
it amazing hand really somehow sleep nothing
sleep somehow hand ,somehow going it running knocking

it going dirty mistake everything
from left running tragedy tomorrow ever
mean tight something life
heart better been face leave i ,else wo hurt step

face leave hurt wo leave boogie hurt where never where
leave sad girl else never late find plase yeah runaway
even ,runaway find plase without almost can done anything
done how almost

- I think the model needs to be trained for still more epochs to get the right semantics, given the time constraint I limited the corpus to 50 songs, so I believe if we train on large corpus of songs we will be able to get even better results.
- We can also look into using better embedding models like Glove embeddings, BERT embeddings to generate semantically better lyrics.